REMARKS

Claims 1-55 were examined. Applicant has amended claims 1 and 49. No claims are cancelled or are newly presented. No new matter has been introduced.

Objections to the drawings:

The drawings are objected to for informalities. Applicant has corrected the drawings.

Objections to the specification:

The examiner objects to the specification because the abstract exceeds 150 words in length and for other informalities. Applicant has amended the specification.

Objections to the claims:

The examiner objects to claims 1 and 49 for minor informalities. Applicant has amended claims 1 and 49. Support for "storing test cases in abstract representations to generate test cases in any target environment script format to provide interoperability between automation tools and cross environment portability of test cases" can be found at least in paragraphs [0002], [0003], [0010], [0011], [0017] and [0018]. Support for "using semantic analysis to decompose test cases into application states, external interaction sequences and input data without changing or deleting an original test case" can be found at least in paragraphs [0034], [0042], [0047] and [0055]. Support for "the test cases being recombined and modified using external rules to combine and modify components of the abstract representation of test cases into new scripts" can be found at least in paragraphs [0010], [0038], [0057], [0060] and [0064].

Rejections under 35 U.S.C. §103

Claims 1-55 stand rejected under §103(a) as anticipated by US 7,313,564 to Melamed et al. in view of "WinRunner 7.0 Tutorial", Mercury Interactive Corp. 2000 (WinRunner).

This ground of rejection is respectively traversed.

In one embodiment of the present invention, as set forth in claim 1, test cases are stored in abstract representations to generate test cases in any target environment script format to provide interoperability between automation tools and cross environment portability of test cases. Semantic analysis is used to decompose test cases into application states, external interaction sequences and input data without changing or deleting an original test case. Rule based test cases are generated using rule based generation of test cases from an abstract representation that includes application states, external interaction sequences and input data of test cases from data stores. The test cases are recombined and modified using external rules to combine and modify components of the abstract representation of test cases into new scripts. Each application state is a set of application objects associated with a set of attributes and their values, or represents a runtime snapshot of an application under test which defines a context of external interaction. The rule based test cases are validated. The rule based test cases are converted to test scripts that are platform independent. The rule based test cases can be used to generate scripts in different target environments.

Test case model: The '564 Patent is focused on GUI objects and based on the GUI map information provided by tools. This is distinguished from the present invention which supports any type of application,

Test case format: In the '564 Patent the format in which test cases are managed is in the final script format of the automation tool used. This is distinguished from the present invention where test cases are stored in abstract representations that are then used to generate test cases in any target environment script format. With the present invention, interoperability between automation tools and cross environment portability of test cases can be achieved.

Test case modification and rules: With The '564 Patent, the modification of test cases is achieved through direct modification of test cases in an editor environment. This is distinguished from the present invention. In the present invention, external rules are used to combine/modify the components of abstract representations of test cases into new scripts. The original test cases and their representation are not changed. As a non-limiting example, consider the case of a 10 character use name field in a test case. With The '564 Patent the test case is modified to test it with different data sets. In each case, it is necessary to specify how to handle success and failure situations. With the present invention, the data entry, normal and exception flows are stored as separate model elements and data is kept separately. Rules are formed to create a test script that takes error data with the data entry test case and uses the exception flow path. The original test case is not changed. A new script is generated based on the rules.

Now, specifically addressing specific sections of the '564 Patent cited by the examiner.

Column 9, line 62, column 10, line 6, column 12, lines 7-14 and lines 38-53, and column 9, line 62 of the '564 Patent specifically refer to test case editing in a specialized test case authoring UI environment.

Column 12, lines 7-14 and lines 38-53 of the '564 Patent refer to representation of UI objects used in testing by specific tools.

Column 13, line 34, column 14, line 49, Figures 1-8, column 12, lines 7-64 of the '564 Patent refer to test scripts being created and sent to a host machine for execution.

With the '564 Patent users use the UI directly and modify the basic test case. This is distinguished from the present invention where rule based test cases are generated, then recombined and modified based on external rules.

In column 14: 34-38 of the '564 Patent, the steps and procedures encoded into the test cases utilize terminology, test strings and/or nomenclature native to an automation tool so that the steps and procedures can be converted into an automation tool recognized test scripts. With the '564 Patent there is no abstract representation of test cases.

With the '564 Patent there is no conversion of original test cases/scripts into an abstract representation. In the '564 Patent the test cases are manipulated in the UI

editor environment manually. This is distinguished from the present invention where test scripts are automatically converted into an abstract representation.

The test case manipulation mentioned in column 9, lines 17-19 and Figure 5 of the '564 Patent is specifically about mapping test cases to requirements/sub-requirements. This is an issue of organizing and managing requirements and test cases together. The present invention is not directed to this area. The present invention is directed to validating test cases in the abstract representation against the application object model. The value of this validation is that the present invention actually finds out which test cases are valid or broken automatically without actually running the test cases or manually verifying it.

Column 14, lines 31-38 of the '564 Patent has nothing to do with validating test cases. This section specifically refers to providing a UI for editing test cases and using the automation tools native scripting language constructs to build the test cases.

Column 13, line 34 and column 14, line 49 of the '564 Patent does not disclose converting rule based test cases to test scripts that are platform independent or that rule based test cases can be used to generate scripts in different target environments. Again, the present invention has rule based generation, which the '564 Patent fails to provide. The present invention uses abstraction representations of test cases, something the '564 Patent fails to teach or suggest Further, the present invention converts original test cases/scripts into abstraction representations. Instead, the '564 patent has a UI editor that manually manipulates test cases.

With WinRunner the GUI map is not an abstract representation. With WinRunner, it is specific to a UI. The WinRunner GUI map represents GUI objects,. The abstract representation of the present invention is a representation of the overall test case, including test steps, states (which includes objects), and test data.

WinRunner creates tool dependency and reduces ability to share the tests. Additionally, WinRunner does not use semantic analysis to convert test cases to abstract representations, and test data is not separated from test steps in abstract representations.

WinRunner supports two modes of development of test scripts: manual script development and recording of user interactions. In WinRunner the tool records user

interactions into a script form which can be played back to execute the tests automatically without manual intervention. One of the disadvantages of this approach is that the scripts generated through recording are too fragile and requires additional manual efforts to ensure reusability for parameterization. Another disadvantage is that the scripts created using one tool will not run on another tool. This reduces the ability to share the test scripts with customer or vendors if they use a different tool.

The present invention specifically addresses the disadvantages of the record and playback tool of WinRunner. With the present invention, the test scripts are converted to a platform independent format (abstract representation) and the test cases can be shared seamlessly. Test scripts can be rebuild from abstract representation for multiple platforms, and the test cases can be executed in any platform from any vendor.

WinRunner uses tool specific scripting as compared to the present invention that uses abstract representation. WinRunner has its own specific scripting language for recording and playback.

This is clearly distinguished from the present invention where test scripts are converted to an abstract representation during the import, and then the test scripts can be generated in multiple languages. we remove tool dependency and increases reuse.

With WinRunner, test scripts contain test data embedded in test steps. Parameterization is done manually to remove data dependency and enable scripts to run with multiple test data. With the present invention, test data is separated from test steps in abstract representations. This makes it easy to run same test cases with multiple data sets.

With WinRunner, application states, external interaction sequences (test steps) and data are mixed together within the scripts. It is very difficult to reuse or modify the test cases later as a result. This is distinguished from the present invention which uses abstract representations to distinguish these entities.

CONCLUSION

It is submitted that the present application is in form for allowance, and such action is respectfully requested. The Commissioner is authorized to charge any additional fees which may be required, including petition fees and extension of time fees, to Deposit Account No. 07-1700 (Docket No. SYM-0004).

Respectfully submitted,
GOODWIN PROCTER LLP

Date: 6.3.08

Paul Davis, Reg. No. 29,294

135 Commonwealth Drive
Menlo Park, CA 94025
Tel: 650/752-3106
Customer No. 77845